

Lecture2

1. Lens flaws

- a. Radial distortion: 画面随着距离中心光轴越远，扭曲越大
- b. vignetting: 画面的四周边缘更暗，边缘的光线更容易在穿过透镜组的时候被挡住
- c. Chromatic aberration: 色彩差异，因为光线在经过透镜的时候发生了折射。
- d. From photon to photo: 因为pixels是按照顺序read in的。所以当遇到快速旋转的螺旋桨时，会因为速度过快而发生扭曲。

lecture4

1. 眼睛结构:

- a. Iris: (虹膜) 控制瞳孔
- b. Pupil: 瞳孔
- c. Lens: 晶状体
- d. Retina: 视网膜

2. 两种不同的photo receptor:

- a. Cone: 末端是三角形的，主要分辨强光下的颜色，对光线不是很敏感。
- b. Rod: 末端是长方形，对光线很敏感，主要在昏暗环境中工作。
- c. 分布：
 - i. Cone集中在视网膜中央
 - ii. Rod分布相对均匀

3. 不同的颜色现实模型：

- a. RGB: 三个channel
 - i. 优点：
 - 1. 简单
 - 2. 常见
 - ii. 缺点：
 - 1. 在RGB空间中的距离远近，和人眼认为的两种颜色的差距不成正比
 - 2. Correlated: 值三个颜色的channel的强度相互影响
- b. HSV: hue & saturation & value
 - i. 优点：
 - 1. 直观的选取颜色
 - 2. 很好转换
 - ii. 缺点：
 - 1. Perceptual non-linearity
 - 2. 对于深颜色缺乏准确度
- c. YCbCr: 亮度+蓝色色差+红色色差
 - i. 优点：
 - 1. 便于压缩和传输
 - ii. 缺点：
 - 1. Perceptual non-linearity
- d. Lab: lightness + green to red + blue to yellow
 - i. 优点：

1. Linear perceptual
- ii. 缺点:
 1. Hard to calculate

4. 光线照到物体表面会发生什么:

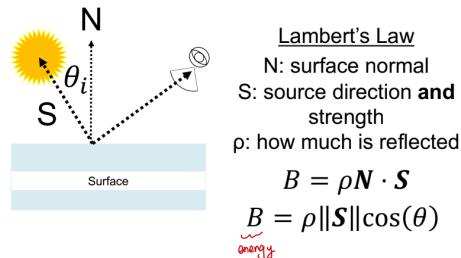
- a. absorbed :吸收并转化为其他能量
- b. transmitted: 透过物体
- c. Reflected: 反射

5. Opaque reflection: 不透明反射

- a. Bi-directional reflectance function (BRDF)
- b. Given the reflected, then use incident angle (入射角) and reflected angle (反射角) 来计算。

6. Specular reflection & diffuse reflection

- a. Diffused reflection:
 - i. it is a Lambertian Surface
 - ii. 反射率和入射角无关, 且均匀反射
 - iii. 计算公式:



- iv. 应用:已知其他, 求N:

$$B_i = \rho N_i \cdot S$$

1D, fixed 2D unknown 3D, unknown

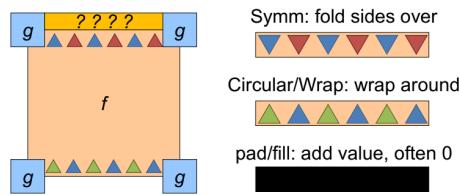
1D unknown

Lecture 5 filter 1

1. Sliding widow method: 有一个kernel, 然后从signal上逐个从左向右滑动, 产出output
2. 处理edge case的方法:

- a. Output的大小选择:
 - i. full : output会比input更大
 - ii. Same: output和input一样大
 - iii. Valid: output比input小
- b. 如何补全input周围的值:
 - i. Symm: 将紧挨的pixel翻转上去
 - ii. circular/wrap: 将对边的pixel平移上来

iii. pad/fill: 自动补全0



3. Linear filter examples:

- a. Average
- b. Shifting
- c. Sharpened

4. Apply filter 的两个性质：

- a. Linear : $\text{apply}(\text{img}, (f_1 + f_2)) = \text{apply}(\text{img}, f_1) + \text{apply}(\text{img}, f_2)$
- b. Shift_invariant: $\text{shift}(\text{apply}(\text{img}, f)) = \text{apply}(\text{shift}(\text{img}), f)$

5. Convolution 和 cross-correlation 的区别

- a. Convolution 需要在计算之前将 kernel flip一下, 而 cross-correlation 则不需要。
- b. Convolution 是 commutative 的, 但 cross-correlation 不是

6. Gaussian 的使用: 避免了 blur 之后很多色块的问题

- a. Sigma 值越大, gaussian 的图像从中心向四周递减越慢。
- b. sigma 的大小和 filter size 之间的关系:
 - i. Kernel size = sigma * 6
- c. Gaussian 可以滤掉高频的 noise 并且保留低频的 signal
- d. 但是 gaussian 也有 fail 的时候: 当有很离谱的 outlier 时, 解决方法--> 使用 median filter
- e. gaussian 的优点是能去除噪点, 但缺点是也会移除细节

7. Median filter

- a. 操作方式: sort filter 范围之内的 input pixel 的值, 找到中间值并作为 output
- b. 是否 linear: 不 linear

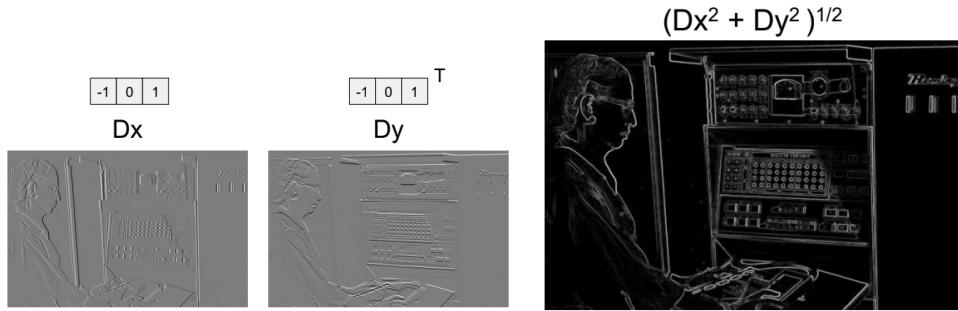
8. Bilateral filter

- Gaussian Filtering
- $I^G(x) = \frac{1}{W} \sum_{x_i \in \Omega} I(x_i) g_\sigma(||x_i - x||)$
- Bilateral Filtering:
 - Weight the kernel based on content similarity!
 - Preserves Edges!
- $I^B(x) = \frac{1}{W} \sum_{x_i \in \Omega} I(x_i) g_\sigma(||x_i - x||) f_\eta(||I(x_i) - I(x)||)$

9. 如何通过得到 detail 进而生成 sharpened 图像:

- a. 首先生成 smooth 的图像
- b. 其次用 original 图像减去 smooth 图像, 进而生成 detailed 图像
- c. 通过选择 alpha 系数值, 将 original 和 detailed 图像进行叠加

10. 通过 Dx 和 Dy 1D filter 来实现对于横向和纵向 edge 的捕捉, 并且可以将 Dx 和 Dy 的值进行结合, 最终生成整个 image 的图像边缘



Lecture 6 detector

1. 使用 3×1 的kernel来计算derivative & prewitt filter & sobel filter

a. 求导原理:

Another one: $\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x+1, y) - f(x-1, y)}{2}$

-1	0	1
----	---	---

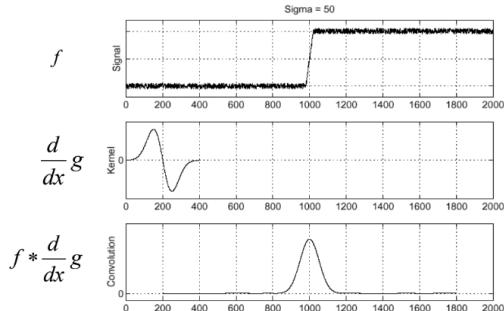
b. Prewitt 和 sobel: 这里的horizontal和vertical指的是filter中数字变化的方向

	Horizontal	Vertical
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

2. 为解决求导后, noise被放大的问题: 先对整个input进行高斯模糊, 之后在进行求导

a. 技巧: 可以不用先求高斯, 再求导。可以通过化简求出一个高斯求导的filter

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$



b. 证明:

Definition of Convolution

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(t) \cdot g(x - t) dt$$

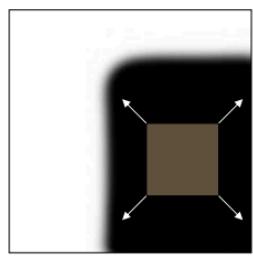
Using Linearity of Differential Operator

$$\frac{d}{dx}(f * g)(x) = \frac{d}{dx} \int_{-\infty}^{+\infty} f(t) \cdot g(x - t) dt = \int_{-\infty}^{+\infty} f(t) \cdot \frac{d}{dx}g(x - t) dt$$

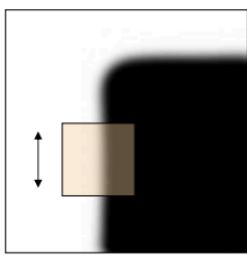
c. Deriv. Of gauss 的缺点:

- i. 非线性
- ii. 计算量大

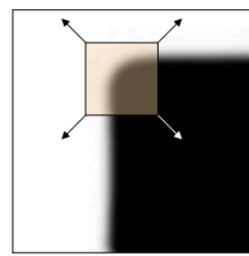
3. Flat & edge & corner 的检测方式:



“flat” region:
no change in
all directions



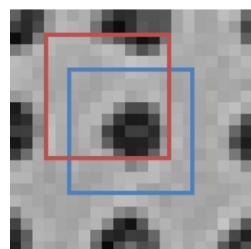
“edge”:
no change
along the edge
direction



“corner”:
significant
change in all
directions

4. 最直观的逻辑:

- a. 选定一个中间window(不变), 然后在这个window的基础上移动得到与之相比较的window。

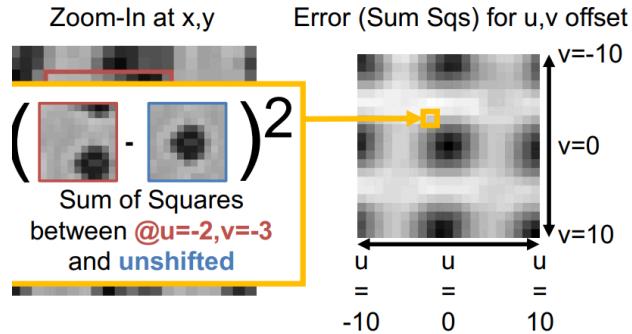


- b. 计算他们直接的error

$$E(u, v) = \sum_{(x,y) \in W} (I[x + u, y + v] - I[x, y])^2$$

( · )^2

c. 遍历所有的u, v值并画出图像



5. 快速的近似方式: 使用 Taylor series

$$f(x + u, y + v) \approx f(x, y) + \frac{\partial f(x, y)}{\partial x} u + \frac{\partial f(x, y)}{\partial y} v$$

a. 则可以通过原始window中的pixel, 一一对应计算出偏转u v之后的像素位置

$$I(x + u, y + v) \approx I(x, y) + I_x u + I_y v$$

b. 证明过程:

$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} (I[x + u, y + v] - I[x, y])^2 \\ &\approx \sum_{(x,y) \in W} (I[x, y] + I_x u + I_y v - I[x, y])^2 \\ &= \sum_{(x,y) \in W} (I_x u + I_y v)^2 \\ &= \sum_{(x,y) \in W} I_x^2 u^2 + 2I_x I_y u v + I_y^2 v^2 \end{aligned}$$

c. 将多项式变成, vector和matrix乘积的形式, 并且将集合符合和matrix相结合

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} (I_x^2 u^2 + 2I_x I_y u v + I_y^2 v^2) \\ &= [u, v] \mathbf{M} [u, v]^T \end{aligned}$$

$$\mathbf{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}$$

i. 假设 $I_x I_y = 0$, 即所有的edge都是水平或者竖直的:

$$\mathbf{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} \approx \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

a,b both small: flat  $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$

One big,
other small: edge  $\begin{bmatrix} 50 & 0 \\ 0 & 0.1 \end{bmatrix}$ or $\begin{bmatrix} 0.1 & 0 \\ 0 & 50 \end{bmatrix}$

a,b both big: corner  $\begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$

ii. 回归到一般情况:

$$\mathbf{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \mathbf{V}^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \mathbf{V}$$

计算方法就是先求出matrix M, 之后对M进行eigenvalue decomposition, 从而得到V(两个eigen vector) 和两个eigenvalue。

Given \mathbf{M} , can decompose it into eigenvectors \mathbf{V} and

eigenvalues λ_1, λ_2 with $\mathbf{M} = \mathbf{V}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{V}$.

λ_1, λ_2 both small: flat 

One big,
other small: edge 

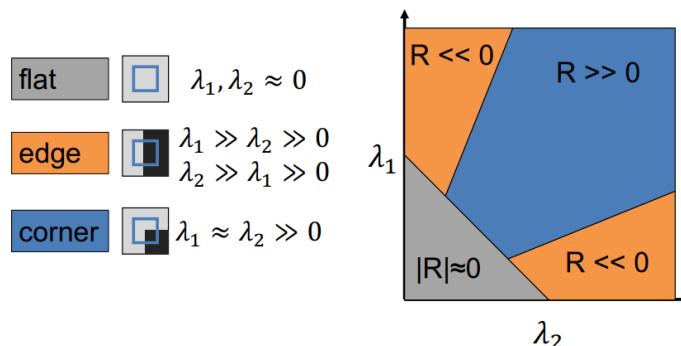
λ_1, λ_2 both big: corner 

d. 为解决之前运算慢的问题, 把求eigenvalues 出来在进行比较的方法变成求R, 用R的大小正负来代表edge, corner, flat

i. R的表达式:

$$R = \det(\mathbf{M}) - \alpha \text{trace}(\mathbf{M})^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

ii. R的大小正负和detector之间的关系:



e. 实际使用:

1. Compute partial derivatives I_x, I_y per pixel
2. Compute \mathbf{M} at each pixel, using Gaussian weighting w

$$\mathbf{M} = \begin{bmatrix} \sum_{x,y \in W} w(x,y) I_x^2 & \sum_{x,y \in W} w(x,y) I_x I_y \\ \sum_{x,y \in W} w(x,y) I_x I_y & \sum_{x,y \in W} w(x,y) I_y^2 \end{bmatrix}$$

3. Compute response function R

$$\begin{aligned} R &= \det(\mathbf{M}) - \alpha \operatorname{trace}(\mathbf{M})^2 \\ &= \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 \end{aligned}$$

4. Threshold R
5. Take only local maxima (called non-maxima suppression)

6. Corner detection 过程中可能遇到的 invariant 和 covariant

a. Covariant:

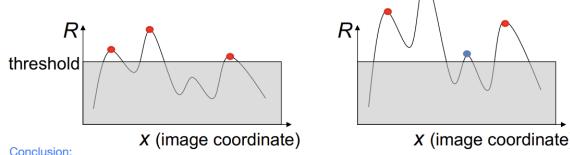
- i. 当光线强度发生变化时: intensity出现了affine change, 在求导后会引入一个常数a, 在threshold不变的情况下会整体拉高peak, 引入新的corner

Affine Intensity Change

$$I_{new} = aI_{old} + b$$

M only depends on derivatives, so b is irrelevant

But a scales derivatives and there's a threshold



Conclusion: Partially invariant to affine intensity changes

- ii. 当image被放大之后, 之前的corner可能变成了一个平滑的曲线。

b. Invariant: (保持不变的两种情况)

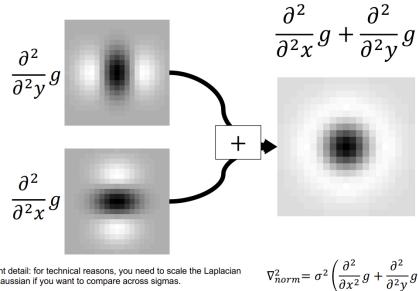
- i. Translation: 什么都不改变
- ii. Rotating: 虽然改变了eigenvector, 因为corner的角度发生了改变, 但是eigenvalue的值没有改变。

Lecture 7 descriptors

1. Blob detection

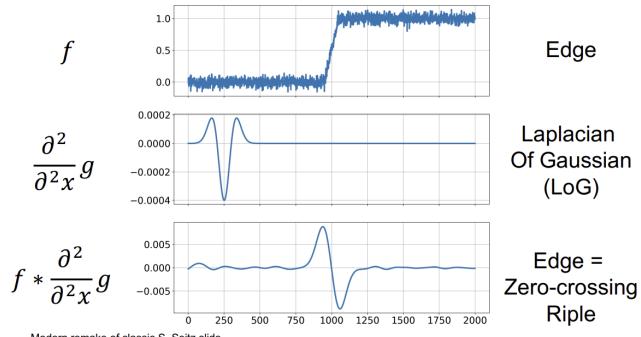
a. 对Gaussian求二导, 并生成新的filter

Laplacian of Gaussian (LoG)

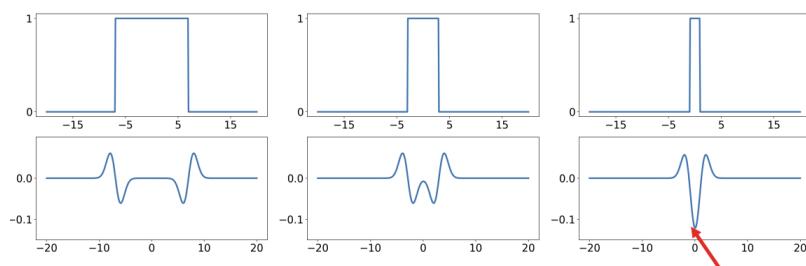


at detail: for technical reasons, you need to scale the Laplacian gaussian if you want to compare across sigmas. $\nabla_{norm}^2 = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial y^2} g \right)$

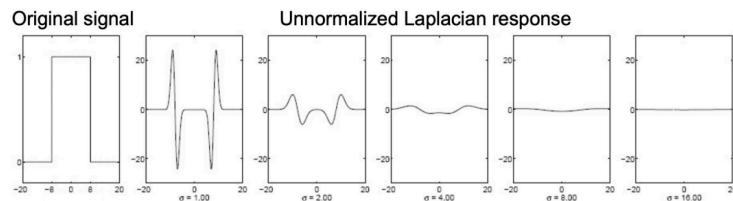
b. 信号处理过程:

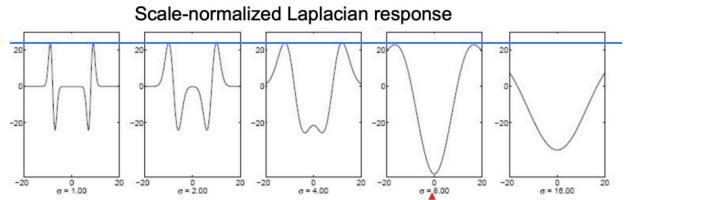


c. 当一个dot 碰到LoG的结果:

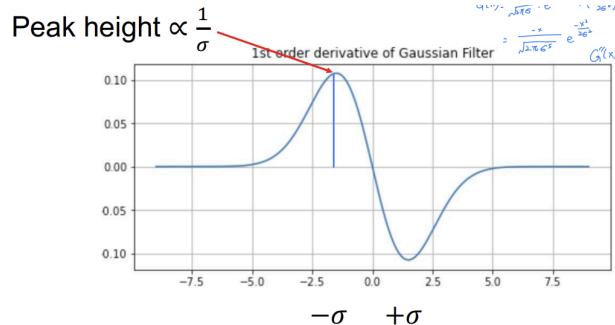


d. 某个大小的dot会和某个sigma值的LoG形成最大的response:但前提是需要normalization处理:





i. 因为peak的大小和 $1/\sigma$ 成正比:



ii. 又因为LoG是二导, 所以要乘以 σ^2 :

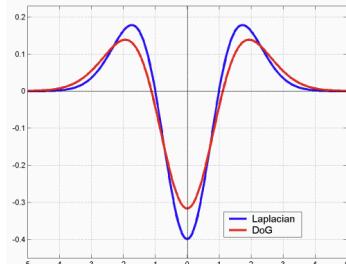
Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

2. 使用Difference of Gaussians 来模拟 Laplacian of Gaussians.

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma)) \quad (\text{Laplacian})$$

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma) \quad (\text{Difference of Gaussians})$$

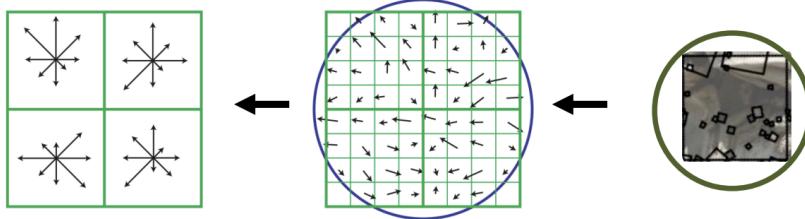
k is empirically determined



3. SIFT:

SIFT Descriptors

All these arrows tell you the gradient direction of each pixel.



1. Compute gradients
2. Build histogram (2x2 shown, 4x4 in practice)
3. (4x4) cells * 8 orientations = 128-dim descriptor

Gradients ignore global illumination changes

4. 2nd nearest neighbor trick:

- a. 使用比值和1之间的大小比较来检测是否match

$$r = \frac{\|\mathbf{x}_q - \mathbf{x}_{1NN}\|}{\|\mathbf{x}_q - \mathbf{x}_{2NN}\|}$$

Lecture 8 transformation1

1. 使用least-squares来表示objective function, 紧接着求导并得出最优的w:

a. 用matrix来表示所有的点:

$$\sum_{i=1}^k (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 \rightarrow \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & 1 \\ \vdots & 1 \\ \mathbf{x}_k & 1 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} m \\ b \end{bmatrix}$$

b. 求导得出最优解:

$$\frac{\partial}{\partial \mathbf{w}} \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|_2^2 = 2\mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{X}^T \mathbf{Y}$$

$$\mathbf{0} = 2\mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{X}^T \mathbf{Y}$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{Y}$$

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

c. 因为objective function可导, 所以就有了两种可能的解法:

- i. 找到closed form然后求导解出w的表达式
- ii. 通过GD, 来梯度下降从而找到最优解:

$$\mathbf{w}_0 = \mathbf{0}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \left(\frac{\partial}{\partial \mathbf{w}} \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|_2^2 \right)$$

2. 如何表达一个line:

a. 使用normal vector n (unit vector) 和 offset d

$$ax + by - d = 0$$

$$\mathbf{n}^T [x, y] - d = 0$$

b. 求解点到直线最短距离:

$$\frac{\mathbf{n}^T [x, y] - d}{\|\mathbf{n}\|_2^2} = \mathbf{n}^T [x, y] - d$$

c. 通过点到直线距离公式, 重新写出objective function:

$$\sum_{i=1}^k (\mathbf{n}^T[\mathbf{x}, \mathbf{y}] - d)^2 \rightarrow \|\mathbf{X}\mathbf{n} - \mathbf{1}d\|_2^2$$

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \mathbf{n} = \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} \quad \boldsymbol{\mu} = \frac{1}{k} \mathbf{1}^T \mathbf{X}$$



d. Mu的用处: mu是vector: x1的平均数, y1的平均数

i. 当我们去尝试求n时会发现 $d=\mu n$

Won't derive, but can show that whenever you find the \mathbf{n} , and d that minimize the objective, $d = \boldsymbol{\mu} \mathbf{n}$.

3. 当我们得到了 $d=\mu n$ 时, 我们就能重新写一下我们的objective function:

$$\begin{aligned} \|\mathbf{X}\mathbf{n} - \mathbf{1}d\|_2^2 &= \|\mathbf{X}\mathbf{n} - \mathbf{1}\boldsymbol{\mu}\mathbf{n}\|_2^2 & d &= \boldsymbol{\mu}\mathbf{n} \\ &= \|(\mathbf{X} - \mathbf{1}\boldsymbol{\mu})\mathbf{n}\|_2^2 \end{aligned}$$

Objective is then:

$$\arg \min_{\substack{\|\mathbf{n}\|=1}} \|(\mathbf{X} - \mathbf{1}\boldsymbol{\mu})\mathbf{n}\|_2^2$$

a. 已知一个线性代数求最值模型, 套用之后通过求eigenvec的方式求得n:

$$\arg \min_{\substack{\|\mathbf{v}\|=1}} \|\mathbf{A}\mathbf{v}\|_2^2 \rightarrow \text{Eigenvector corresponding to smallest eigenvalue of } \mathbf{A}^T \mathbf{A}$$

b. 运用到我们的objective function中就变成了:

$$\mathbf{n} = \text{smallest_eigenvec}((\mathbf{X} - \mathbf{1}\boldsymbol{\mu})^T(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}))$$

$$(\mathbf{X} - \mathbf{1}\boldsymbol{\mu})^T(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}) = \begin{bmatrix} \sum_i (x_i - \mu_x)^2 & \sum_i (x_i - \mu_x)(y_i - \mu_y) \\ \sum_i (x_i - \mu_x)(y_i - \mu_y) & \sum_i (y_i - \mu_y)^2 \end{bmatrix}$$

4. RANSAC(random sample consensus)

a. Algorithm:

- i. 随便选两个点形成一根线
- ii. 根据这根线计算所有点的least squares loss
- iii. 和之前的最优解进行对比, 如果loss更小, 则更新最优解

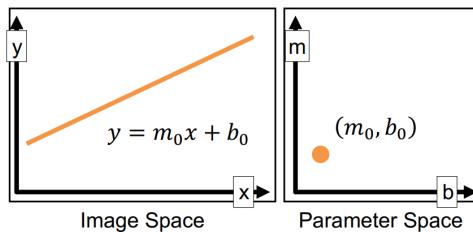
b. 可以设置一个threshold, 当在这个区间内的, 就算正确判断, 最后比较不同模型判断正确的点的数量

c. 如何计算在给定trials的前提下, 选取到没有outliers的几率:

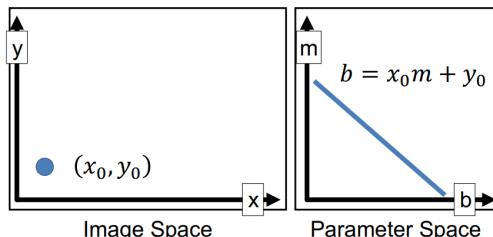
- i. 失败: 所有的pair全是outlier
- ii. 成功: 至少有一个pair中包含一个inliner

5. Hough Transform:

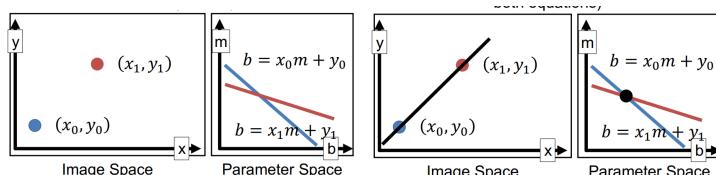
a. Image-space 中的点 \rightarrow parameter space 中的线：



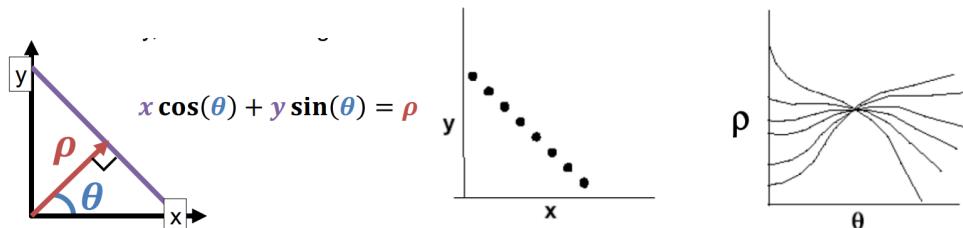
b. image -space 中的线 \rightarrow parameter space中的点：



c. 如何确定image space中的两个点组成的线：



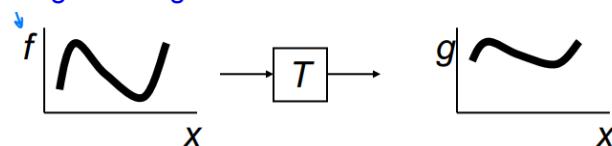
d. M 和 b的模型存在诸多不足, 于是改为sita和rou的模型：



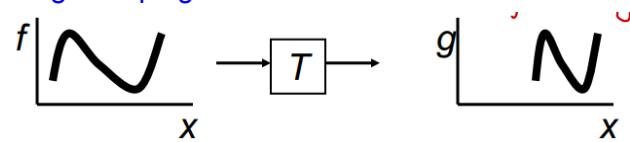
Lecture9 translation2

1. Image transformation 的分类.

a. Image filtering :



b. Image warping:

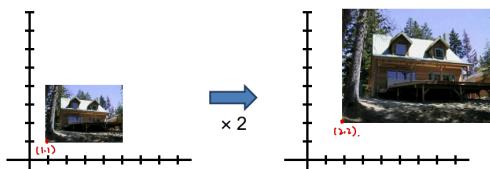


2. Parametric (global) warping 的分类.

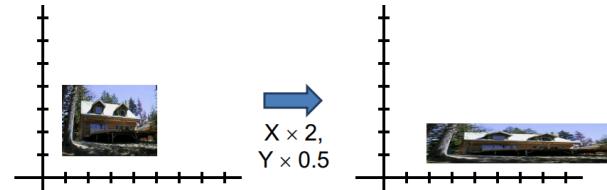
- a. Translation
- b. Rotation
- c. Aspect
- d. Affine
- e. Perspective
- f. Cylindrical

3. Scaling

- a. Uniform scaling: 直接给原始坐标x,y乘上一个同一个系数得到新的坐标:



- b. Non-uniform scaling: 对于原始坐标x, y 乘以不同的系数:



- c. 所形成的matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

4. Rotation:

- a. 所需要的matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- b. 是否线性:

x' is a linear combination/function of x, y
 x' is not a linear function of θ

5. Shear 将矩形变成平行四边形

- a. Matix we need here:



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

6. Projection 都preserve了什么 vs 2*2 matrix又preserve了什么

- a. Projection:

- i. 直线还是直线
 - ii. 平行线不再是平行线
 - iii. 事物的大小变化
- b. 2*2matrix:
- i. 直线还是直线
 - ii. 平行线还是平行线
 - iii. 原点还是原点

7. translation: 不能通过2*2的matrix来实现

- a. 需要3*3的matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} s_x & 0 & a \\ 0 & s_y & b \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

B. Affine transformation:

- a. 3*3matrix由transformation和translation构成:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

↑ transformation matrix ↑ translation operation
 2x2 linear transformation matrix

使用homogeneous

$$x' = Ax + b \quad \text{正常笛卡尔坐标}$$

- b. Affine transformation前后什么是保持不变的, 什么是发生变化的
- i. 直线还是直线
 - ii. 平行线还是平行线
 - iii. 但是原点不再是原点

9. 如何将平移、旋转、变换相结合

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{T(t_x, t_y)} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta)} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S(s_x, s_y)}$$

translation rotation scaling

10. Perspective transformations: 不再让最后一行是001

- a. Matrix的结构:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- b. 有几个自由度: 8个 \rightarrow 可以让 $\|H\| = 1$ 或者让 $i=1$
- c. Perspective transformation之后, 什么是不变的, 什么是变化的
 - i. 线还是线
 - ii. 原点不再是原点
 - iii. 平行线不再是平行线

11. Fitting transformation:

- a. 定义: 两个图片, 根据两张图片上的特征点, 进而求出所需要的 affine transformation的参数

- b. Input & model & objective function:

Data: (x_i, y_i, x'_i, y'_i) for $i=1, \dots, k$

Model:
 $[x'_i, y'_i] = M[x_i, y_i] + t$

Objective function:
 $\|[x'_i, y'_i] - (M[x_i, y_i] + t)\|^2$

- c.

Given correspondences: $[x'_i, y'_i] \leftrightarrow [x_i, y_i]$

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Set up two equations per point

$$\begin{array}{c} \text{Up: } 2k \quad \text{Down: } 6 \\ \left[\begin{array}{c} \vdots \\ x'_i \\ y'_i \\ \vdots \end{array} \right] = \left[\begin{array}{cccccc} x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ \vdots & & & & & \end{array} \right] \left[\begin{array}{c} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{array} \right] \\ \boxed{\mathbf{b}} \quad \boxed{\mathbf{A}} \quad \boxed{\mathbf{x}} \end{array}$$

- d. 因为equation过多而导致over constrained, 所以我们无法直接解出parameters, 我们得使用求最小值的方法:

Want: $\mathbf{b} = \mathbf{Ax}$ (\mathbf{x} contains all parameters)

Overconstrained, so solve $\arg \min \|A\mathbf{x} - \mathbf{b}\|$

- e. 当使用homographies时的情况:

I define $x_i^* = x_i / w$.

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i^*/w \\ y_i^*/w \\ 1 \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i \approx \frac{ax_i + by_i + c}{gx_i + hy_i + i}$$

$$y'_i \approx \frac{dx_i + ey_i + f}{gx_i + hy_i + i}$$

$$x_i'(gx_i + hy_i + i) = ax_i + by_i + c$$

$$y_i'(gx_i + hy_i + i) = dx_i + ey_i + f$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'i \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1'i \\ \vdots & & & & & & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A: $2k \times 9$
 $h: 9$

f. 如何求h的问题

1. Eigenvalue decomposition of $A^T A$
2. Singular Value Decomposition (SVD)

g. 为了解出h, 我们最少需要四个点。

12. 步骤:

Step 1: Find “features” to match

Step 2: Describe Features

Step 3: Match by Nearest Neighbor

Step 4: Fit H via RANSAC

Step 5: Blend Images

Lecture10 linear model

1. ML 问题的分类:

	Supervised (Data+Labels)	Unsupervised (Just Data)
Discrete Output	Classification/ Categorization	Clustering
Continuous Output	Regression	Dimensionality Reduction

2. 线性回归问题中的Least squares:

a. Variable数量和equation数量之间的关系:

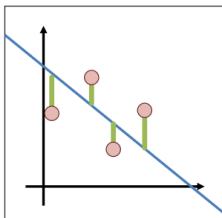
- N eqns, <N vars = overdetermined (will have errors)
- N eqns, N vars = exact solution
- N eqns, >N vars = underdetermined (infinite solns)

b. 训练模型的方法

Data: $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$

Model: (m, b) $y_i = mx_i + b$
Or (w) $y_i = w^T x_i$

Objective function:
 $(y_i - w^T x_i)^2$



c. 转换成matrix calculation之后的式子：

$$\sum_{i=1}^k (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \rightarrow \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

3. Feature 和 data point的数量

- a. Data point比feature数量多，可以正常求inverse
- b. Data point 比feature数量少，会产生rank deficient，会导致matrix的乘积无法inverse，会导致infinite solution。
- c. 为了解决模型的feature过多，使用regularization来解决

4. Regularized Least Squares

a. 在objective function 中添加新的term

Before: $\arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \longrightarrow \text{Loss}$

After: $\arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$

Loss Trade-off Regularization

b. 新的W的解

$$\mathbf{w}^* = \underbrace{(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1}}_{\text{Matrix}} \mathbf{X}^T \mathbf{y}$$

c. 引入validation data来挑选最合适的lambda

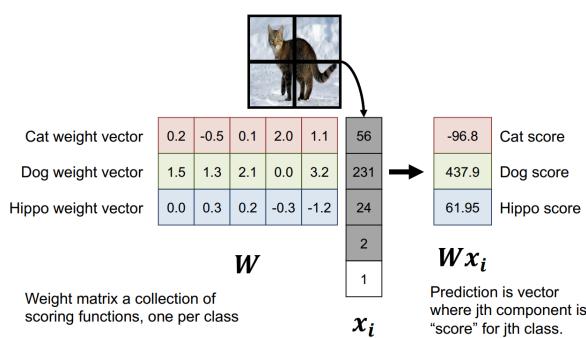
5. 如何做classification

- a. Classification by Least-Squares：先用很多cat的照片训练这个线性回归模型，然后得到一个W，之后再用这个W和input的imageX 相乘，用0.5和结果相比。

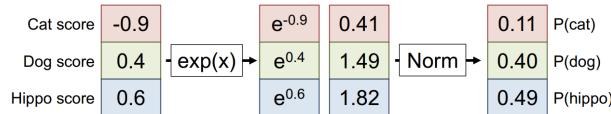
Model – one weight per class: $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$

$\mathbf{w}_0^T \mathbf{x}$ big if cat
 $\mathbf{w}_1^T \mathbf{x}$ big if dog
 $\mathbf{w}_2^T \mathbf{x}$ big if hippo

Stack together: $\mathbf{W}_{3 \times F}$ where \mathbf{x} is in \mathbb{R}^F



- i. Converting Scores to “Probability Distribution”



ii. softmax:

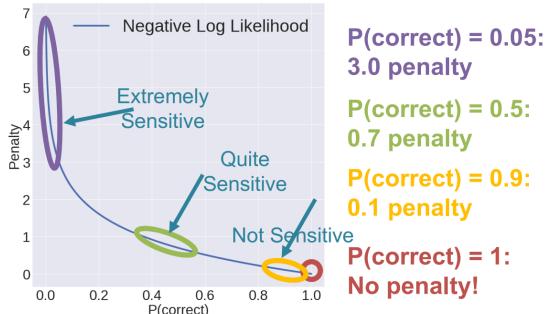
$$\frac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$$

iii. 训练方式:

Training (\mathbf{x}_i, y_i) :

$$\arg \min_{\mathbf{W}} \lambda \|\mathbf{W}\|_2^2 + \sum_i^n -\log \left(\underbrace{\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}}_{\text{Pay penalty for negative log-likelihood of correct class}} \right)$$

Regularization
Over all data points



iv. 为什么使用Negative Log Softmax: 因为我们要惩罚那些对正确的label自信成都很低的, 约不自信, 惩罚强度越大。

b. 最简单的方式: 存储一个cat的照片, 如果跟这个一模一样就是, 否则就不是:

- i. 改进方法: 计算test image和已知的几个类型的image之间的distance, 最终找到距离最近的label。(Nearest Neighbor)
- ii. K-Nearest Neighbors: 找到和test data距离最近的K个label, 然后对这些结果进行平均。--> 用training data建立起模型, 用validation data 进行K值的选择

Lecture11 optimization

1. 找寻最低点的方法:

a. 使用蛮力:

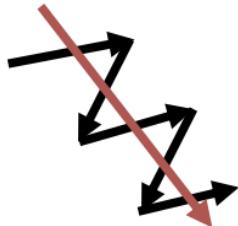
- i. Grid search: 算出区域内每个点的值, 直接相互比较, 找到最小值
- ii. Random search: 随机取点, 计算出对应的值, 最后相互比较。
 - 1. 缺点: 当在高维空间时, 点十分分散, 会导致取样不足

b. 使用gradient::

- i. 箭头的方向, 表示值增大的方向

- ii. subgradient: 指的是当objective function 在某一点不可导, 则我们就通过预先定义该点倒数的方法来解决该问题。
- iii. Gradient 的分类:
 - 1. GD (vanilla) :每次更新, 需要平均所有data的gradient
 - 2. SGD:每次更新, 只根据一个点的gradient
 - 3. minibatch GD:每次更新, 平均一个minibatch中的datapoint
- iv. 每次找到的minimum会因为起始点的不同而不同

- c. 使用momentum:average gradients



$$m_t = \beta \cdot m_{t-1} + (1 - \beta) \nabla_w L(\mathbf{w})$$

Use m_t instead of $\nabla_w L(\mathbf{w})$

2. Overfitting and underfitting

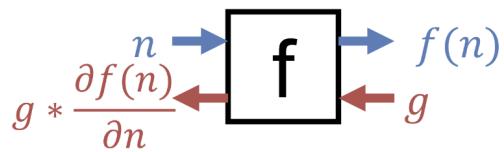
- a. Underfitting:
 - i. 模型variable 过少, 不能很好的代表dataset
 - ii. Bias很高(模型本身造成的错误)
- b. Overfitting:
 - i. 模型variable过多, 导致过度fit training dataset, 这会导致在test dataset中表现的很差
 - ii. Variance很高, training dataset中的datapoint有一点变化都会引起model本身发生很大变化
- c. 解决方法:引入regularization从而惩罚复杂的模型
 - i. 模型的parameter数量相同但是
- d. 相同复杂度的模型, training dataset越小, 越容易overfitting

Lecture12 backpropagation and neural nets

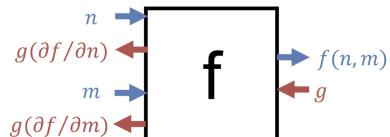
1. Forward and backwards

- a. 当input和output都是一个的时候

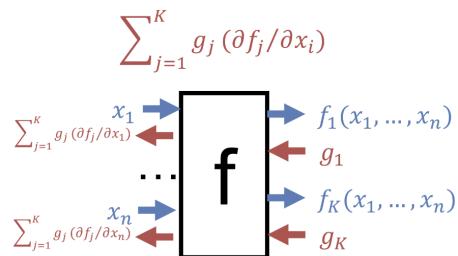
Forward: Given forward input n , compute $f(n)$
Backwards: Given backwards input g , return $g * \frac{\partial f(n)}{\partial n}$



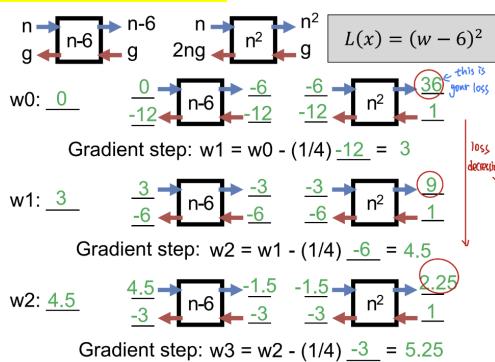
b. 当input是多个, output是一个时:



c. 当input是多个, output也是多个时:

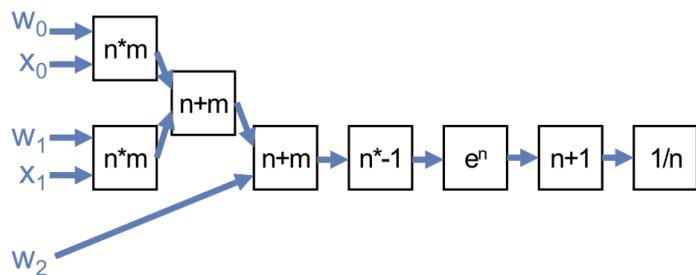


2. 带有数字的例子:



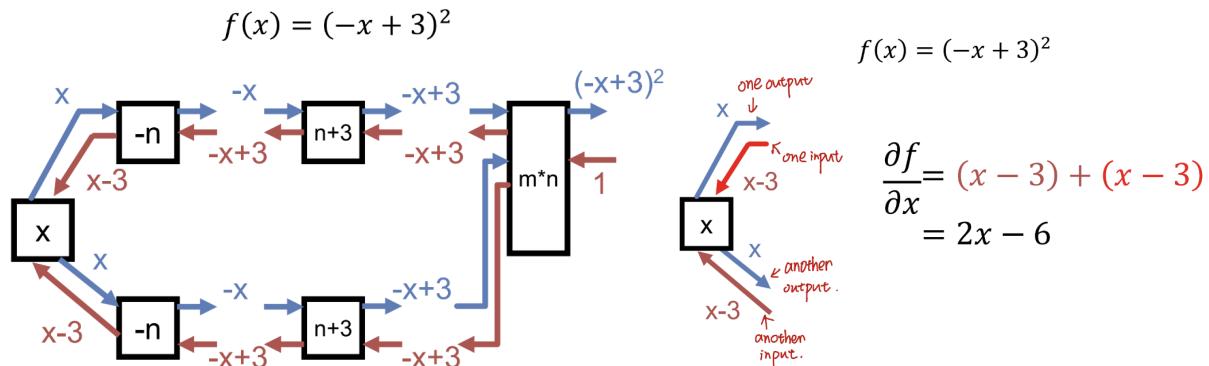
3. 如何将一个复杂的式子, 转换成diagram的形式:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

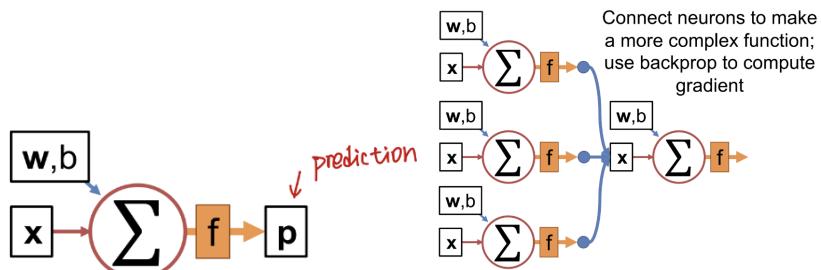


4. 如何解决一个box的output同时用于两个分支的计算:

Multiple Outputs Flowing Back

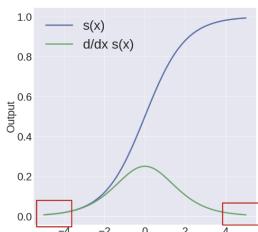


5. 一个独立的artificial neuron的样子:



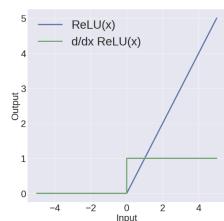
6. Active function 的分类:

a. sigmoid: $s(x) = \frac{1}{1 + e^{-x}}$



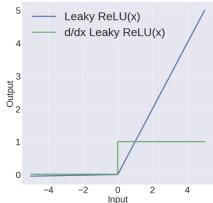
- i. 缺点:当input太大或太小时, output基本为零, 意味着这个神经元处于dead状态

b. ReLU: $\max(0, x)$



$$x : x \geq 0$$

c. Leaky ReLU: $0.01x : x < 0$

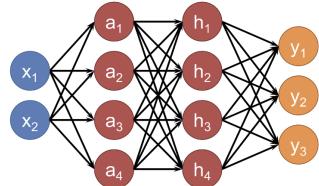


i. 优点: 对于负值可能有反应, really small gradient

7. Fully connected Network:

a. 结构:

Input Hidden 1 Hidden 2 Output



$$\mathbf{h} = f(\mathbf{W}\mathbf{a} + \mathbf{b}) = f\left(\begin{array}{c} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{array}\right) = f\left(\begin{array}{c} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \mathbf{w}_4^T \end{array}\right) \begin{array}{c} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{array} + \begin{array}{c} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{array}\right)$$

Lecture13 CNN1

1. 如何计算fully connected network所需要的parameter

a. 组成部分: weight + bias



Weights: $3 \times 4 + 4 \times 4 + 4 \times 2 = 36$

Parameters: $36 + 11 = 47$

b. Fc 的缺点:

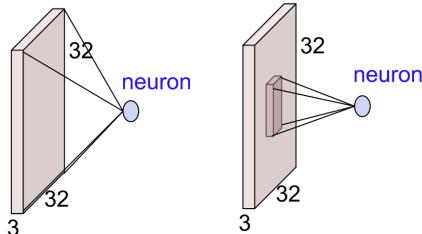
- i. 需要很多parameter
- ii. 破坏了image原有的空间结构

2. Convnet:

a. 和fc的对比:

Fully connected:
Connects to everything

Convnet:
Connects locally
Filter Small & Reused

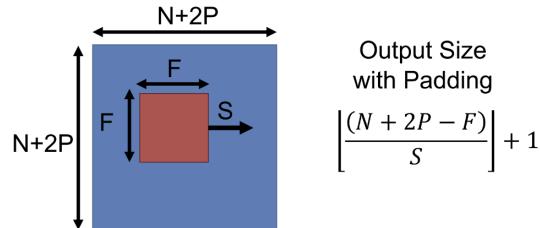


b. Filter 的数量可以大于1

c. 关于filter移动的parameters:

- i. Stride: 每次filter移动几格
- ii. Padding: 原始图像单边往外补偿几格
- iii. N: 原始图像的边长(默认正方形)
- iv. F: filter的宽度

d. 计算filter convolution之后的size公式:



3. Pooling:

- a. 定义: 就是降维, 原本 $2*2$ 的matrix, 通过pooling直接变成一个
- b. 种类:
 - i. Max pooling: 从中找最大值
 - ii. Average pooling: 求平均值

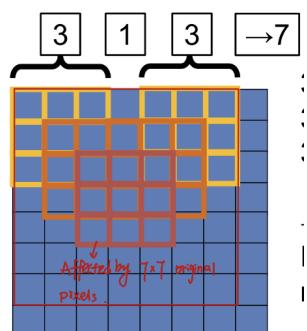
Lecture 14 CNN2

1. 如何将 $H*W*C$ 的volume $\rightarrow 1*1*F$ 的vector:

- a. 用pooling, 前提是 $C == F$, 若 $C \neq F$, 则还需要通过FC将C转化成F
- b. Concolve: 选择 $H*W*C$ 的filter, 则通过input和filter的convolution可以得到一个值, 然后我们准备F个这样的filter就能最终得到 $1*1*F$ 的vector了

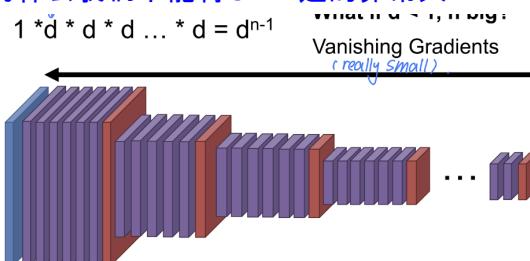
2. Filter size的选择:

- a. 可以直接选则 $7*7$ 或者 重复使用三次 $3*3$ 的filter

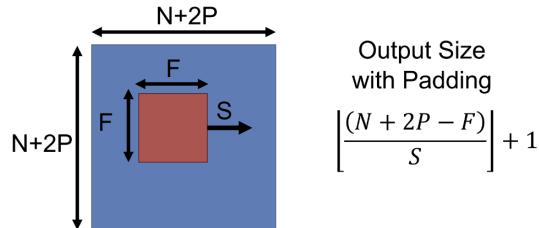


- b. 选择重复三次 $3*3$ filter的好处: 使用可更少的parameter
 - i. $7*7$ filter使用了: 49 parameter 一次ReLU
 - ii. 重复三次 $3*3$ filter: $9 * 3 = 27$ parameter 三次ReLU
- c. 我们希望引入跟多的non-linearity 比如 ReLU

3. 为什么我们不能将CNN建的非常大:



- a. 原因: 因为当我们做backpropofation的时候, 需要从后往前依次乘以每层的gradient。如果grandient小于一的话, 最终的backpropogation结果基本为零。而如果每层的gradient大于零的话, 又会是的最终的结果巨大。



3. Pooling:

- a. 定义: 就是降维, 原本 $2*2$ 的matrix, 通过pooling直接变成一个
- b. 种类:
 - i. Max pooling: 从中找最大值
 - ii. Average pooling: 求平均值

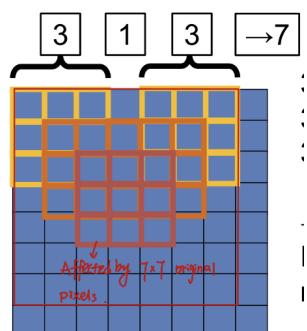
Lecture 14 CNN2

1. 如何将 $H*W*C$ 的volume $\rightarrow 1*1*F$ 的vector:

- a. 用pooling, 前提是 $C == F$, 若 $C \neq F$, 则还需要通过FC将C转化成F
- b. Concolve: 选择 $H*W*C$ 的filter, 则通过input和filter的convolution可以得到一个值, 然后我们准备F个这样的filter就能最终得到 $1*1*F$ 的vector了

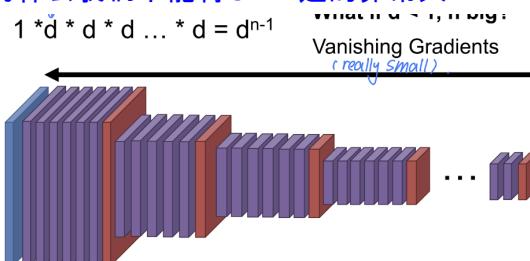
2. Filter size的选择:

- a. 可以直接选则 $7*7$ 或者 重复使用三次 $3*3$ 的filter



- b. 选择重复三次 $3*3$ filter的好处: 使用可更少的parameter
 - i. $7*7$ filter使用了: 49 parameter 一次ReLU
 - ii. 重复三次 $3*3$ filter: $9 * 3 = 27$ parameter 三次ReLU
- c. 我们希望引入跟多的non-linearity 比如 ReLU

3. 为什么我们不能将CNN建的非常大:



- a. 原因: 因为当我们做backpropagation的时候, 需要从后往前依次乘以每层的gradient。如果gradient小于一的话, 最终的backpropagation结果基本为零。而如果每层的gradient大于零的话, 又会是的最终的结果巨大。